Short report for the paper

**Boyuan Chen**

Clip from paper abstract:

“In this work, we propose an alternative superpixel segmentation method based on Gaussian mixture model (GMM) by assuming that each superpixel corresponds to a Gaussian distribution, and assuming that each pixel is generated by first randomly choosing one distribution from several Gaussian distributions which are defined to be related to that pixel, and then the pixel is drawn from the selected distribution. Based on this assumption, each pixel is supposed to be drawn from a mixture of Gaussian distributions with unknown parameters (GMM). An algorithm based on expectation-maximization method is applied to estimate the unknown parameters. Once the unknown parameters are obtained, the superpixel label of a pixel is determined by a posterior probability. The success of applying GMM to superpixel segmentation depends on the two major differences between the traditional GMM-based clustering and the proposed one: data points in our model may be nonidentically distributed, and we present an approach to control the shape of the estimated Gaussian functions by adjusting their covariance matrices.”

**Code running:**

I ran the code with “shenzhouyu.jpg” in the Images folder. And you can see the outputs in result folder with varying values for v\_x and v\_y.

If you open the code file (gmmsp\_demo.m), you would see that v\_x and v\_y are like the variances for each gaussian component. The bigger their values are, the bigger each segment area is and thus the smaller total number of segments is.

**How it works:**

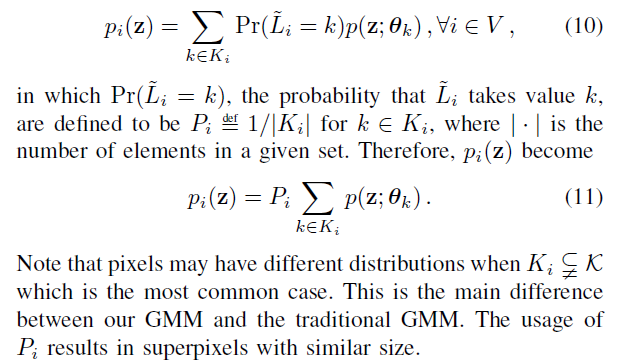
A. General Skeleton

This model is different in that:

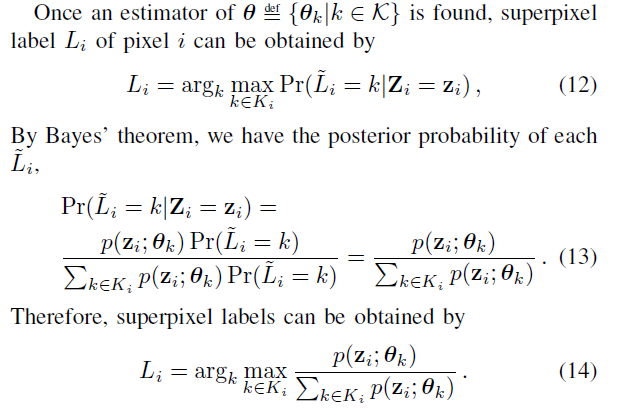
1. It assumes each cluster(z) has the same possibility, and thus the posts are the same

2. Pixels would not consider the clusters that are too far from them. Thus, the distribution of each pixel is different.

Then, the calculation is straightforward:



(10 and 11 are just adding up the possibilities that a pixel is in each of the Gaussian components)



B. The complicated for loop that finds Σ and μ and updates R

These are what they would do after they find the θ. So how do they find it? By MLE, using Jensen’s inequality. Each step we calculate the Σ and μ that maximizes the expectation of L(θ); then, we do it again to find a better R, Σ and μ, until the for loop reaches its end. This is basically the same as the usual approach.

C. Modifying covariance matrix Σ

Sometimes Σ might not be invertible. To cope with this issue, the team replaces the 0 eigenvalues with some small constants. The constants you use here determine the smoothness of the final boundaries.